AFRL-AFOSR-VA-TR-2016-0306

Integrating Programming Language and Operating System Information Security Mechanisms

Stephen Chong
HARVARD COLLEGE PRESIDENT & FELLOWS OF
1350 MASS AVE STE 600
CAMBRIDGE, MA 02138-3846

09/03/2016
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/RTA2

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
31-08-2016	Final	01-06-2013 - 31-05-2016

4. TITLE AND SUBTITLE
Integrating Programming Language and Operating System Information Security Mechanisms

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-12-1-0262

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Stephen Chong

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Harvard University
1350 Mass Ave
Cambridge MA 02138

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Office of Scientific Research
875 North Randolph Street
Suite 325, Room 3112
Arlington VA, 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION A: Distribution approved for public release.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This grant aims to improve the guarantees offered by both language-based information security mechanisms, and operating system information security mechanisms. It seeks to do so by investigating interactions between language-based and OS mechanisms for information security, and exploiting these interactions both to improve the precision of security enforcement, and to provide greater assurance of information security.

This grant focuses on two key projects: language-based control of authority; and formal guarantees for the correctness of audit information.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE		6	Stephen Chong
					19b. TELEPHONE NUMBER *(Include area code)* 617 496-6382

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

AFOSR Final Report

Grant Title:	**Integrating Programming Language and Operating System Information Security Mechanisms**
Grant Number:	**FA9550-12-1-0262**
PI:	**Stephen Chong**
	chong@seas.harvard.edu
Organization	**Harvard University**
Program Manager	**Tristan Nguyen**
Dates covered	**June 1 2013–May 31 2016**

Abstract

This grant aims to improve the guarantees offered by both language-based information security mechanisms, and operating system information security mechanisms. It seeks to do so by investigating interactions between language-based and OS mechanisms for information security, and exploiting these interactions both to improve the precision of security enforcement, and to provide greater assurance of information security.

This grant focuses on two key projects: language-based control of authority; and formal guarantees for the correctness of audit information.

Highlights of the reporting period:

- Design, implementation, and release of Shill, a secure shell scripting language. See http://shill-lang.org/.
- Design and implementation an extensible framework for authority control, capable of expressing and composing many existing and novel access control mechanisms.
- Introduced formal definition for the correctness of audit logs, and designed and implemented an approach to declare audit policies and automatically ensure that correct audit logs are generated during program execution.
- Explored the use of declarative policies on capabilities to ensure correct usage, including access-control and information-flow policies that restrict propagation and use of capabilities.
- Seven peer-reviewed publications, including one journal article, and five in top security and programming language conferences.
    1. Moore, S., C. Dimoulas, M. Flatt, R. B. Findler, and S. Chong (2016, October). Extensible access control with authorization contracts. In *Proceedings of the 29th Annual ACM SIGPLAN Conference on Object-*

Oriented Programming Languages, Systems, Languages, and Applications, New York, NY, USA. ACM Press. To appear.
2. Amir-Mohammadian, S., S. Chong, and C. Skalka (2016, April). Correct audit logging: Theory and practice. In *5th International Conference on Principles of Security and Trust*.
3. Chong, S. and R. van der Meyden (2015, December). Using architecture to reason about information security. *ACM Transactions on Information and System Security* 18(2).
4. Askarov, A., S. Moore, C. Dimoulas, and S. Chong (2015, July). Cryptographic enforcement of language-based erasure. In *Proceedings of the 28th IEEE Computer Security Foundations Symposium*, Piscataway, NJ, USA. IEEE Press.
5. Johnson, A., L. Waye, S. Moore, and S. Chong (2015, June). Exploring and enforcing security guarantees via program dependence graphs. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, New York, NY, USA, pp. 291–302. ACM Press.
6. Moore, S., C. Dimoulas, D. King, and S. Chong (2014, October). Shill: A secure shell scripting language. In *11th USENIX Symposium on Operating Systems Design and Implementation*. USENIX.
7. Dimoulas, C., S. Moore, A. Askarov, and S. Chong (2014, June). Declarative policies for capability control. In *Proceedings of the 27th IEEE Computer Security Foundations Symposium*, Piscataway, NJ, USA. IEEE Press.

- Support of two graduate students, an undergrad, and a postdoctoral fellow
  - Including one PhD dissertation ("Software Contracts for Security", by Scott Moore) and one senior thesis.
- Outreach
  - Many talks (at least 6 by PI Chong) on Shill and related research, including at Cornell University, NII Shonan (Japan), and Brown University.
  - Postdoctoral Fellow Christos Dimoulas attended a 2014 Dagstuhl Seminar on "Scripting Languages and Frameworks: Analysis and Verification", and presented work on Shill.
  - Graduate student Scott Moore attended the 2014 Vail Computer Elements Workshop (VCEW) and presented preliminary work on Shill.
- Shill publicly released, available via http://shill-lang.org/.
- Provisional patent granted on core technology underlying Shill.
- Awarded a Physical Sciences and Engineering Accelerator grant by Harvard University to help develop the technology underlying Shill.

Participants

Stephen Chong (PI)
Christos Dimoulas (Postdoctoral research fellow)
Scott Moore (Graduate student)

Daniel King (Graduate student)
Daniel Bradley (Undergraduate)

Collaborators
Christian Skalka (Associate Professor, University of Vermont)

Introduction
The objective of this grant is to improve the guarantees offered by both language-based information security mechanisms, and operating system information security mechanisms. It seeks to do so by investigating interactions between language-based and OS mechanisms for information security, and exploiting these interactions both to improve the precision of security enforcement, and to provide greater assurance of information security.

Language-based information security uses programming language abstractions and techniques to reason about and enforce information security. Language-level abstractions and mechanisms can provide strong fine-grained application-specific information security guarantees. By contrast, operating system (OS) information security mechanisms use OS-level abstractions to provide isolation and protection for processes executing in a system; recent operating system mechanisms can provide fine-grained isolation and protection.

Synergies between programming language and OS mechanisms provide opportunity to improve information security guarantees in at least two ways: (1) increasing the precision of operating-system mechanisms; and (2) improving the assurance of language-based mechanisms.

The sponsored work has focused on two key projects: language-based control of authority, and reliable auditing. In the remainder of the report, we describe progress in the last year on these two projects, and then provide a summary of the research performed over the lifetime of this grant.

Language-based control of authority
The Principle of Least Privilege suggests that software should be executed with no more authority than it requires to accomplish its task. Current security tools make it difficult to apply this principle: they either require significant modifications to applications or do not facilitate reasoning about combining untrustworthy components.

We have explored using programming language techniques to specify and enforce restrictions on the authority of components. Previously under this award, we explored declarative policies to restrict the use of capabilities, and designed Shill, a secure shell scripting language. In this last year of the project, we focused both on

extending the usability of Shill, and also exploring the foundations of authority control. For the usability of Shill, we have started a port of Shill to the Linux operating system, which will greatly enhance the applicability of the tool.

In addition, we have received a provisional patent ("Method For End-To-End Enforcement Of Security Policies In A Scripting Language", Application number 62243900) for some of the key concepts embodied in Shill.

Investigation of the foundations of authority control lead to the recognition that existing programming language access control frameworks do not meet the needs of all software components, and the development of an expressive framework to implement access control monitors for components. The basis of the framework is a novel concept: the *authority environment*. An authority environment associates rights with an execution context. The building blocks of access control monitors in our framework are *authorization contracts*: software contracts that manage authority environments. We implemented a diverse set of existing access control mechanisms and writing custom access control monitors for three realistic case studies, demonstrating the expressiveness and applicability of the framework. This work will be published at OOPSLA 2016, a top-tier programming language conference.

In addition, this award has supported additional relevant research, including the use of application architecture to enforce high-level application-specific information security guarantees (Chong and van der Meyden, 2015), the use of cryptography to enforce expressive information security policies (Askarov et al., 2015), and sophisticated program analysis techniques to discover and enforce application-specific security guarantees (Johnson et al., 2015).

Summary of Shill

Shill scripts enable compositional reasoning about security through contracts that limit the effects of script execution, including the effects of programs invoked by the script. Shill contracts are declarative security policies that act as documentation for consumers of Shill scripts, and are enforced through a combination of language design and sandboxing.

In work under this grant in previous years, we implemented a prototype of Shill for FreeBSD. We have been developing a Linux version of Shill.

Shill uses declarative security policies that describe and limit the effects of script execution, including effects of arbitrary programs invoked by the script.
These declarative security policies can be used by producers of software to provide fine-grained descriptions of the authority the software needs to execute. This, in turn, allows consumers of software to inspect the software's required authority, and make an informed decision to execute the software, reject the software, or apply a more restrictive policy on the software. The Shill runtime system ensures that script

execution adheres to the declared security policy, providing a simple mechanism to restrict the authority of software.

Two key features enable Shill declarative security policies: language-level capabilities and contracts. Shill scripts access system resources only through capabilities: unforgeable tokens that confer privileges on resources. Shill scripts receive capabilities only from the script invoker; Shill scripts cannot store or arbitrarily create capabilities. Moreover, Shill uses capability-based sandboxes to control the execution of arbitrary software. Thus, the capabilities that a user passes to a Shill script limit the script's authority, including any programs it invokes. Shill's contracts specify what capabilities a script requires and how it intends to use them. Shill's runtime and sandboxes enforce these contracts, hence they serve as fine-grained, expressive, declarative security policies that bound the effects of a script.

Personnel

This grant has supported postdoctoral research fellow Christos Dimoulas (who joined the project in January 2013) and graduate student Scott Moore (who joined the project in Fall 2013). Both Dimoulas and Moore this year have been working on the design and implementation of Shill, and formal foundations for the control of authority in computer systems. Moore graduated in Summer 2016, and is continuing to develop the technology underlying Shill.

Integration of language-level and OS mechanisms for provenance

Provenance is the history of computation. Audit logs are a form of provenance, as are execution traces, and meta-data such as version information recorded by a version-control system, or timestamp and ownership information recorded by a file system. Auditing underlies retroactive security frameworks, and has become increasingly important to the theory and practice of cybersecurity.

In systems where auditing is used, programs are typically instrumented to generate audit logs, based on some formal or informal auditing policy. However, even if auditing policies are formal, it is difficult to ensure that execution of manually instrumented programs will generate a "correct" audit log and guarantee expected accountability.

In this project, we investigate language-level mechanisms for specifying and enforcing audit policies, i.e., specifying what information should be recorded in an audit log, and automatically instrumenting the program to provably capture that information.

In collaboration with Christian Skalka (UVM), we have developed a novel semantics of auditing based on information algebra, along with proof techniques for ensuring correctness of automated program instrumentation strategies, aka retrofitting. We

have defined a retrofitting strategy that supports a general class of auditing policies, and proved that this retrofitting strategy is guaranteed to correctly enforce a general class of user-specified auditing policies. Moreover, we have implemented this model for the Java programming language, and applied it to audit an open-source medical records application. This work appeared at the 5th International Conference on Principles of Security and Trust (POST), in April 2016.

Personnel

We are collaborating with Christian Skalka (UVM) on this work.

AFOSR Deliverables Submission Survey

Response ID:6778 Data

1.

Report Type

Final Report

Primary Contact Email

Contact email if there is a problem with the report.

chong@seas.harvard.edu

Primary Contact Phone Number

Contact phone number if there is a problem with the report

617 496-6382

Organization / Institution name

Harvard University

Grant/Contract Title

The full title of the funded effort.

Integrating Programming Language and Operating System Information Security Mechanisms

Grant/Contract Number

AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-12-1-0262

Principal Investigator Name

The full name of the principal investigator on the grant or contract.

Stephen Chong

Program Officer

The AFOSR Program Officer currently assigned to the award

Tristan Nguyen

Reporting Period Start Date

06/01/2013

Reporting Period End Date

05/31/2016

Abstract

This grant aims to improve the guarantees offered by both language-based information security mechanisms, and operating system information security mechanisms. It seeks to do so by investigating interactions between language-based and OS mechanisms for information security, and exploiting these interactions both to improve the precision of security enforcement, and to provide greater assurance of information security.

This grant focuses on two key projects: language-based control of authority; and formal guarantees for the correctness of audit information.

Highlights of the reporting period:
* Design, implementation, and release of Shill, a secure shell scripting language. See http://shill-lang.org/.
* Design and implementation an extensible framework for authority control, capable of expressing and composing many existing and novel access control mechanisms.
* Introduced formal definition for the correctness of audit logs, and designed and implemented an approach

to declare audit policies and automatically ensure that correct audit logs are generated during program execution.
* Explored the use of declarative policies on capabilities to ensure correct usage, including access-control and information-flow policies that restrict propagation and use of capabilities.
* Seven peer-reviewed publications, including one journal article, and five in top security and programming language conferences.
* Support of two graduate students, an undergrad, and a postdoctoral fellow, including one PhD dissertation ("Software Contracts for Security", by Scott Moore) and one senior thesis.
Outreach
- Many talks (at least 6 by PI Chong) on Shill and related research, including at Cornell University, NII Shonan (Japan), and Brown University.
- Postdoctoral Fellow Christos Dimoulas attended a 2014 Dagstuhl Seminar on "Scripting Languages and Frameworks: Analysis and Verification", and presented work on Shill.
- Graduate student Scott Moore attended the 2014 Vail Computer Elements Workshop (VCEW) and presented preliminary work on Shill.
* Shill publicly released, available via http://shill-lang.org/.
* Provisional patent granted on core technology underlying Shill.
* Awarded a Physical Sciences and Engineering Accelerator grant by Harvard University to help develop the technology underlying Shill.

Distribution Statement

This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement

If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF The maximum file size for an SF298 is 50MB.

sf0298.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The maximum file size for the Report Document is 50MB.

Chong AFOSR YIP Final report.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

Moore, S., C. Dimoulas, M. Flatt, R. B. Findler, and S. Chong (2016, October). Extensible access control with authorization contracts. In Proceedings of the 29th Annual ACM SIGPLAN Conference on Object-Oriented Programming Languages, Systems, Languages, and Applications, New York, NY, USA. ACM Press. To appear.

Amir-Mohammadian, S., S. Chong, and C. Skalka (2016, April). Correct audit logging: Theory and practice. In 5th International Conference on Principles of Security and Trust.

Chong, S. and R. van der Meyden (2015, December). Using architecture to reason about information security. ACM Transactions on Information and System Security 18(2).

Askarov, A., S. Moore, C. Dimoulas, and S. Chong (2015, July). Cryptographic enforcement of language-based erasure. In Proceedings of the 28th IEEE Computer Security Foundations Symposium, Piscataway, NJ, USA. IEEE Press.

Johnson, A., L. Waye, S. Moore, and S. Chong (2015, June). Exploring and enforcing security guarantees via program dependence graphs. In Proceedings of the 36th ACM SIGPLAN Conference on Programming

Language Design and Implementation, New York, NY, USA, pp. 291–302. ACM Press.

Moore, S., C. Dimoulas, D. King, and S. Chong (2014, October). Shill: A secure shell scripting language. In 11th USENIX Symposium on Operating Systems Design and Implementation. USENIX.

Dimoulas, C., S. Moore, A. Askarov, and S. Chong (2014, June). Declarative policies for capability control. In Proceedings of the 27th IEEE Computer Security Foundations Symposium, Piscataway, NJ, USA. IEEE Press.

New discoveries, inventions, or patent disclosures:

Do you have any discoveries, inventions, or patent disclosures to report for this period?

Yes

Please describe and include any notable dates

Received provisional patent "Method For End-To-End Enforcement Of Security Policies In A Scripting Language", application number 62243900, received 20-OCT-2015.

Do you plan to pursue a claim for personal or organizational intellectual property?

Yes

Changes in research objectives (if any):

Change in AFOSR Program Officer, if any:

Initial program officer: Robert Herklotz
Current program officer: Tristan Nguyen

Extensions granted or milestones slipped, if any:

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Aug 31, 2016 18:41:42 Success: Email Sent to: chong@seas.harvard.edu